Лабораториска вежба 2

Имплементација на поедноставена верзија на Kerberos протоколот

Прво направив AES класа која ги содржи методите за енкрипција и декрипција, со кои пораките меѓу засегнатите страни соодветно се енкриптираат и декриптираат.

```
public  String AES_encrypt(String strToEncrypt, String secret)
{
    try
    {
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        return java.util.Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.getBytes( charsetName: "UTF
    }
    catch (Exception e)
    {
        System.out.println("Error while encrypting: " + e.toString());
    }
    return null;
}

public  String AES_decrypt(String strToDecrypt, String secret)
{
    try
    {
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        return new String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
    }
    catch (Exception e)
    {
        System.out.println("Error while decrypting: " + e.toString());
    }
    return null;
}
```

```
SecretKeySpec secretKey;
byte[] key;

public  void setKey(String myKey)
{
    MessageDigest sha = null;
    try {
        key = myKey.getBytes( charsetName: "UTF-8");
        sha = MessageDigest.getInstance("SHA-1");
        key = sha.digest(key);
        key = Arrays.copyOf(key,  newLength: 16);
        secretKey = new SecretKeySpec(key,  algorithm: "AES");
    }
    catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
    }
    catch (UnsupportedEncodingException e)  {
        e.printStackTrace();
    }
}
```

Јана Ѓелевска 171084

Во прилог е кодот на класата Alice во која се чуваат информациите за нејзиниот ID во IDa, нејзиниот клуч Ka, на Bob (со оној кој сака да комуницира) и при креирање на објект од оваа класа се содавава nonce така што се хешира вредноста на клучот на Alice.

```
}
class Alice{

    String IDa;
    String IDb;
    String Ka;
    byte[] nonceRa;

    public Alice(String IDa,String IDb,String key) throws UnsupportedEncodingException, NoSuchAlgorithmException, Invali
        this.IDa=IDa;
        this.IDb=IDb;
        this.Ka=key;
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        this.nonceRa=Arrays.copyOf(md.digest(this.Ka.getBytes( charsetName: "UTF-8")), newLength: 16);

    }
```

Во методата verification како параметар е даден YaYb кој претставува ticket генериран од KDC, со помош на AES се декриптира пораката Ya пратена од KDC и потоа соодветно се дели на делови кои ќе се верифицираат. Прво се запишува сесискиот клуч, потоа се проверува вредноста на nonce дали се совпаѓа со онаа на Alice, истото се прави и за IDb, и се проверува дали дадениот lifetimeT е истечен или не. На крај го генерира својот timestamp кој заедно со IDa се енкриптираат со сесискиот клуч .

```
    public String verification(String yaYb) throws IllegalBlockSizeException, InvalidKeyException, NoSuchPaddingExceptio
        System.out.println("-----Alice-----");
        String ya=yaYb.split( regex: ";")[0];
        //dekripcija na ya
        AES aestmp= new AES();
        System.out.println("Ka: "+Ka);
        String decText=aestmp.AES_decrypt(ya,Ka);
        System.out.println("Posle dekripcija Ya: "+decText);
        String Kses=decText.substring(0,16);
        System.out.println("Kses: "+Kses);
        //verify ra
        String ra=decText.substring(decText.length()-38,decText.length()-24);
        System.out.println("nonceRa:" +new String(nonceRa));
        System.out.println("nonce od Ya: "+ra);
        if (!ra.equals(new String(this.nonceRa)))
        {
            throw new InvalidParameterException("Nonce");
        }
        //verify IDb
        String Idb=decText.substring(decText.length()-16);
        System.out.println("IDb:" +IDb);
        System.out.println("IBb od Ya: "+Idb);
        if (!Idb.equals(this.IDb))
        {
            throw new InvalidParameterException("IDb");
        }
        String []T=decText.substring(decText.length()-24,decText.length()-16).split( regex: ";");
        Time time=new Time(Integer.parseInt(T[0]),Integer.parseInt(T[1]),Integer.parseInt(T[2]));
        Time Ts=new Time(System.currentTimeMillis());
        System.out.println("T od Ya: " +time.toString());
        System.out.println("Ts: "+Ts.toString());
        if(!Ts.after(time))
        {
            throw new InvalidParameterException("T");
        }
        //encrypt yab=ekses(IDa,Ts)
        String plainText=IDa+Ts.toString();
        String Yab=aestmp.AES_encrypt(plainText,Kses);
        System.out.println("Alice response : "+Yab+yaYb.split( regex: ";")[1]);
        return Yab+";"+yaYb.split( regex: ";")[1];
```

Во прилог е кодот за класата KDC која ги чува клучевите на страните кои ќе комуницираат Ka на Alice и Kb на Bob.

```
}
class KDC{
    String Ka;
    String Kb;
    public KDC(String ka, String kb)
    {
        this.Kb=kb;
        this.Ka=ka;
    }
```

Методот ticketGranting како параметри ги зема ID Alice IDa ID Bob IDb и nonce на Alice, овие податоци му ги праќа Alice во вид на Request. Прво се генерира сесиски клуч Kses, кој ќе се користи за оваа сесија(комуникација во одреден период), со помош на класата SecureRandom и истиот е 16 бајти. Потоа со помош на класата Time се генерира lifetime T на сесијата, добиен со додавање 3 часа на моменталното време. На крај се врши енкрипција на на сесискиот клуч Kses, nonce, lifetime T и IDb со клучот на Alice, излезот е во променливата Ya и енкрипција на сесискиот клуч Kses, IDa и lifetime со сесискиот клуч во променливата Yb. Како излез функцијата враќа String во кој се конкатенирани YaYb.

```
    public String ticketGranting(String IDa, String IDb, byte[] ra) throws IllegalBlockSizeException, InvalidKeyExceptio
        System.out.println("-----KDC-----");
        //generate random Kses
        Random randomGen=new SecureRandom();
        byte[] Kses=new byte[16];
        randomGen.nextBytes(Kses);
        //generate lifetime T
        Time T=new Time(System.currentTimeMillis()+3600000*3);
        //ya encryption with Ka (Kses,ra,T,IDb)
        AES aes=new AES();
        StringBuilder sb= new StringBuilder();
        //kses 16
        //T 8
        //ra 14
        //id 16
        sb.append(new String(Kses)).append(new String(ra)).append(T.toString()).append(IDb);
        System.out.println("Ka: "+Ka);
        System.out.println("Pred enkripcija Ya: "+sb.toString());
        String ya=aes.AES_encrypt(sb.toString(),Ka);
        //yb encryption with Kb (kses ,IDa,T)
        sb=new StringBuilder();
        sb.append(new String(Kses)).append(IDa).append(T.toString());
        System.out.println("Kb: "+Kb);
        System.out.println("Pred enkripcija Yb "+sb.toString());
        String yb=aes.AES_encrypt(sb.toString(),Kb);
        System.out.println("KDC response: "+ya+yb);
        return ya+";"+yb;

    }
```

Во класата Bob се чува неговотоID- IDb, ID на оној со кој сака да комуницира IDa и неговиот клуч Kb.

```
class Bob{
    String IDb;
    String IDa;
    String Kb;

    public Bob(String IDb,String IDa,String kb) throws UnsupportedEncodingException {
        this.IDa=IDa;
        this.IDb=IDb;
        this.Kb=kb;
    }
```

Во методата verification како параметар е даден YabYb кој претставува порака пратена од Alice, со помош на AES се декриптира пораката Yb со помош на клучот на Bob и потоа соодветно се дели на делови кои ќе се верифицираат, се запишува сесискиот клуч Kses и се проверува дали сеуште е валидна сесијата T lifetime. Потоа се декриптира и другиот дел Yab со помош на сесискиот клуч, се проверува дали IDa се совпаѓа со она кое го чува Bob и се проверува и Ts.

```
    public String verification(String yabYb) throws IllegalBlockSizeException, InvalidKeyException, NoSuchPaddingExcepti
        System.out.println("-----Bob-----");
        //Yb dekripcija
        String yb=yabYb.split( regex: ";")[1];
        AES aestmp=new AES();
        String decText=aestmp.AES_decrypt(yb,this.Kb);
        System.out.println("Posle dekripcija Yb: "+decText);
        //Kses
        String Kses=decText.substring(0,16);
        System.out.println("Kses: " + Kses);
        //yab dekripcija
        String yab=yabYb.split( regex: ";")[0];
        String decTextYab=aestmp.AES_decrypt(yab,Kses);
        System.out.println("Posle dekripcija Yab: "+decTextYab);
        //verify IDa
        String ida=decTextYab.substring(decTextYab.length()-24,decTextYab.length()-8);
        System.out.println("IDa: "+IDa);
        System.out.println("IDa od Yab: "+ida);
        if (!ida.equals(this.IDa))
        {
            throw new InvalidParameterException("IDa");
        }
        //verify lifetime T

        String []T=decText.substring(decText.length()-8).split( regex: ":");
        Time time=new Time(Integer.parseInt(T[0]),Integer.parseInt(T[1]),Integer.parseInt(T[2]));
        System.out.println("T: " +time.toString());
        Time current=new Time(System.currentTimeMillis());
        if(!current.after(time))
        {
            throw new InvalidParameterException("T");
        }
        //verify timestamp Ts
        String []T1=decTextYab.substring(decTextYab.length()-8).split( regex: ":");
        Time Ts=new Time(Integer.parseInt(T1[0]),Integer.parseInt(T1[1]),Integer.parseInt(T1[2]));
        System.out.println("Ts: " +Ts.toString());
        if(!current.after(Ts))
        {
            throw new InvalidParameterException("Ts");
        }
```

Класата Kerberos служи за одвивање на протоколот, прво се зададени вредности за клучевите и ID на страните кои ќе комуницираат и за сите 3 страни во протоколот се креираат објекти од нивните класи. Потоа се повикува функцијата ticketGranting од KDC во која влезни параметри се земаат од Alice (наместо request) и која генерира енкриптирана порака како одговор. Тој одговор е влезен параметар за функцијата verification повикана од Alice која доколку е успешна како одговор враќа енкриптирана порака за Bob. На крај Bobja повикува својата функција verification и соодветно ги проверува и декриптира податоците.

```
public class Kerberos {

    public static void main(String[] args) throws UnsupportedEncodingException, InvalidKeyException, BadPaddingException

        String ka="kriptografija123";
        String kb="bezbednostkluc12";
        String IDa="janajanajanajana";
        String IDb="bobobobobobobobo";
        Alice alice= new Alice(IDa,IDb,ka);
        Bob bob=new Bob(IDb,IDa,kb);
        KDC kdc=new KDC(ka,kb);
        //alice prakja request
        String response=kdc.ticketGranting(alice.IDa,alice.IDb,alice.nonceRa);
        //alice vreificira i enkriptira
        String aliceVer=alice.verification(response);
        //Bob verificira
        String bobVer=bob.verification(aliceVer);
```

Во секоја функција се печатат вредностите кои се пред енкрипција и по декрипција за да може да се воочи протоколот, излезот од оваа програма е на следната слика.

Линк за кодот:

https://github.com/GJJana/BNKS_Labs

Јана Ѓелевска 171084